

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ
компьютерной безопасности и
криптографии

Идеалы полугрупп

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студентки 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Шуликиной Анастасии Александровны

Преподаватель

профессор, д.ф.-м.н.

подпись, дата

В. А. Молчанов

Саратов 2022

СОДЕРЖАНИЕ

| | |
|---|---|
| ВВЕДЕНИЕ | 3 |
| 1 Цель работы и порядок её выполнения | 4 |
| 2 Теория | 5 |
| 2.1 Понятия идеалов полугрупп | 5 |
| 2.1.1 Алгоритм построения идеалов полугруппы по таблице Кэли | 6 |
| 2.2 Понятия и свойства отношений Грина на полугруппах | 6 |
| 2.2.1 Алгоритм вычисления отношений Грина и построения «egg-box»-картины | 7 |
| 2.3 Код программы, на основе рассмотренных алгоритмов, на языке C++ | 9 |

ВВЕДЕНИЕ

В данной лабораторной работе поставлена задача рассмотрения понятия идеалов полугрупп, разбор и реализация алгоритмов их построения, понятия и свойства отношений Грина на полугруппах, разбор и реализация алгоритмов вычисления отношений Грина и построения «egg-box»-картины конечной полугруппы.

1 Цель работы и порядок её выполнения

Цель работы – изучение строения полугрупп с помощью отношений Грина.

Порядок выполнения работы:

1. Рассмотреть понятия идеалов полугруппы. Разработать алгоритмы построения идеалов полугруппы по таблице Кэли.
2. Рассмотреть понятия и свойства отношений Грина на полугруппах.
3. Разработать алгоритмы вычисления отношений Грина и построения «egg-box»-картины конечной полугруппы.

2 Теория

2.1 Понятия идеалов полугрупп

Полугруппа – это алгебра $S = (S, \cdot)$ с одной ассоциативной бинарной операцией \cdot , т.е. выполняется $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ для любых $x, y, z \in S$.

Если полугрупповая операция называется умножением (соответственно, сложением), то полугруппу называют мультипликативной (соответственно, аддитивной).

Пусть S – произвольная полугруппа. Непустое подмножество $I \subset S$ называется правым (соответственно, левым) идеалом полугруппы S , если для любых $x \in I, y \in S$ выполняется условие: $xy \in I$ (соответственно $yx \in I$), т.е. $I \cdot S \subset I$ (соответственно, $S \cdot I \subset I$). Если I – одновременно левый и правый идеал полугруппы S , то I называется двусторонним идеалом (или просто идеалом) полугруппы S . Ясно, что в коммутативной полугруппе S все эти определения совпадают.

Лемма 1. Множество всех идеалов IdS (соответственно, левых идеалов $LIdS$ или правых идеалов $RIdS$) любой полугруппы S является системой замыкания.

Пусть X – подмножество полугруппы S . Тогда наименьший правый идеал полугруппы S , содержащий подмножество X , равен $(X] = XS^1 = X \cup XS$, наименьший левый идеал полугруппы S , содержащий подмножество X , равен $[X) = S^1X = X \cup SX$ наименьший идеал полугруппы S , содержащий подмножество X , равен $[X] = S^1XS^1 = X \cup XS \cup SX \cup SXS$.

В частности, любой элемент $a \in S$ определяет наименьшие правый, левый и двусторонний идеалы: $(a] = aS^1, [a) = S^1a$ и $[a] = S^1aS^1$, которые называются главными (соответственно, правыми, левыми и двусторонними) идеалами.

Минимальные относительно теоретико-множественного включения идеалы (соответственно, левые или правые идеалы) называются минимальными идеалами (соответственно, минимальными левыми или правыми идеалами).

Лемма 2. Если полугруппа имеет минимальный идеал, то он является ее наименьшим идеалом и называется ядром полугруппы.

Доказательство. Если I – минимальный идеал полугруппы S , то для любого идеала J полугруппы S непустое множество $IJ \subset I \cap J \subset I$ и, значит, идеал $I \cap J = I, I \subset J$.

Любая конечная полугруппа имеет наименьший идеал, т.е. ядро полугруппы.

Доказательство. Для конечной полугруппы S множество всех идеалов IdS конечно и, значит, его пересечение является наименьшим идеалом S .

2.1.1 Алгоритм построения идеалов полугруппы по таблице Кэли

Вход. Полугруппа S , таблица Кэли размерностью N , выполняющая свойство ассоциативности.

Выход. Множество правых идеалов R , множество левых идеалов L и множество двусторонних идеалов I .

Шаг 1. Строится множество res , состоящее из всех возможных комбинаций элементов полугруппы S (сочетание без повторений): $res = \{\{S_1\}, \{S_2\}, \dots, \{S_N\}, \dots, \{S_1, S_2\}, \{S_1, S_3\}, \dots, \{S_1, S_N\}, \dots, \{S_2, S_3\}, \dots, \{S_2, S_N\}, \dots, \{S_1, S_2, S_N\}\}$.

Шаг 2. Цикл i от 1 по N .

Шаг 2.1. Проверяем все подмножества множества res на выполнение условия правого идеала: если $\forall res_i \in res : xy \in res_i \forall x \in res_i, y \in S$, если условие выполняется, то res_i добавлем в множество R .

Шаг 2.2. Проверяем все подмножества множества res на выполнение условия левого идеала: если $\forall res_i \in res : yx \in res_i \forall x \in res_i, y \in S$, если условие выполняется, то res_i добавлем в множество L .

Шаг 2.3. Для того, чтобы множество res_i являлось двусторонним идеалом, оно должно удовлетворять условия правого и левого идеала. Если все подмножества множества res выполняют эти условия, то res_i добавляем в I .

Шаг 3. Выводится R, L, I .

Трудоемкость алгоритма $O(N^3 * M * M_2)$, M - размер множества res , M_2 - размер множества res_i .

2.2 Понятия и свойства отношений Грина на полугруппах

Отображения $f : a \mapsto [a]$, $f_r : a \mapsto (a)$, $f_l : a \mapsto [a]$, $a \in S$ определяют ядра $\mathcal{J} = \ker f$, $\mathcal{R} = \ker f_r$, $\mathcal{L} = \ker f_l$ по формулам:

$$(a, b) \in \mathcal{J} \iff [a] = [b],$$

$$(a, b) \in \mathcal{R} \iff (a) = (b),$$

$$(a, b) \in \mathcal{L} \iff [a] = [b].$$

Все эти отношения, а также отношения $\mathcal{D} = \mathcal{R} \vee \mathcal{L}$, $\mathcal{H} = \mathcal{R} \cap \mathcal{L}$ являются эквивалентностями на множестве S , которые называются отношениями Грина полугруппы S . Классы этих эквивалентностей, порожденные элементом $a \in S$, обозначаются J_a , R_a , L_a , D_a и H_a , соответственно.

Лемма. Отношения Грина полугруппы S удовлетворяют следующим свойствам:

1. эквивалентность \mathcal{R} регулярна слева и эквивалентность \mathcal{L} регулярна справа, т.е. $(a, b) \in \mathcal{R} \Rightarrow (xa, xb) \in \mathcal{R}$ и $(a, b) \in \mathcal{L} \Rightarrow (ax, bx) \in \mathcal{L}$ для любых $x \in S$,
2. эквивалентности \mathcal{R} , \mathcal{L} коммутируют,
3. $\mathcal{D} = \mathcal{R} \cdot \mathcal{L} = \mathcal{L} \cdot \mathcal{R}$,
4. если полугруппа S конечна, то $\mathcal{D} = \mathcal{I}$,
5. любой класс D эквивалентности \mathcal{D} можно изобразить с помощью следующей egg-box-диаграммы, клетки которой являются классами эквивалентности \mathcal{H} , лежащими в D .

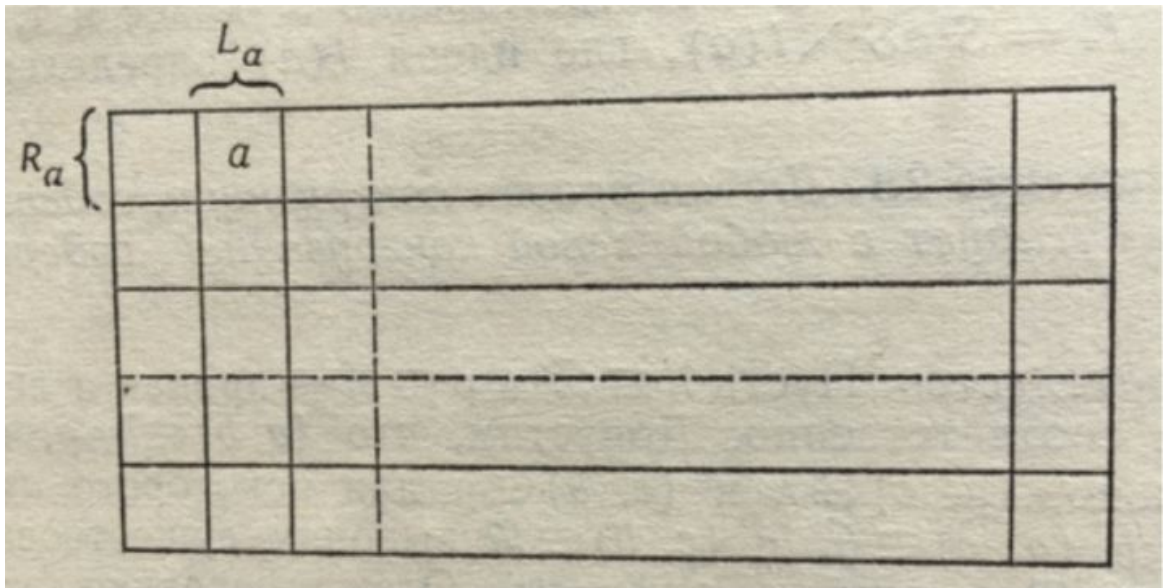


Рисунок 1 – egg-box-диаграмма

2.2.1 Алгоритм вычисления отношений Грина и построения «egg-box»-картины

Вход. Полугруппа S , таблица Кэли размерностью N , выполняющая свойство ассоциативности.

Выход. Отношения Грина \mathcal{R} , \mathcal{L} , \mathcal{J} , \mathcal{H} , \mathcal{D} и «egg-box»-картины.

Шаг 1. Создаем булеву переменную $chek = true$.

Шаг 2. Цикл i от 1 до N .

Шаг 2.1. Для каждого i цикл j от 1 до N .

Шаг 2.2. $\forall S_i, S_j \in S$: строим правые идеалы $(S_i], (S_j]$, если $(S_i] = (S_j]$, то добавляем (S_i, S_j) в множество \mathcal{R} .

Шаг 2.3. $\forall S_i, S_j \in S$: строим левые идеалы $[S_i), [S_j)$, если $[S_i) = [S_j)$, то добавляем (S_i, S_j) в множество \mathcal{L} .

Шаг 2.4. $\forall S_i, S_j \in S$: строим двусторонние идеалы $[S_i], [S_j]$, если $[S_i) = [S_j)$, то добавляем (S_i, S_j) в множество \mathcal{J} .

Шаг 2.5. Множество \mathcal{H} строится: $\mathcal{H} = \mathcal{R} \cap \mathcal{L}$.

Шаг 2.6. Множество \mathcal{D} строится $\mathcal{D} = \mathcal{R} \cup \mathcal{L}$.

Шаг 3. Цикл по k от 0 до $D.size$.

Шаг 3.1. Проверяются все классы эквивалентности \mathcal{R} , если они совпадают с k -ым элементом множества \mathcal{D} , то они добавляются в $res1$.

Шаг 3.2. Проверяются все классы эквивалентности \mathcal{L} , если они совпадают с k -ым элементом множества \mathcal{D} , то они добавляются в $res2$.

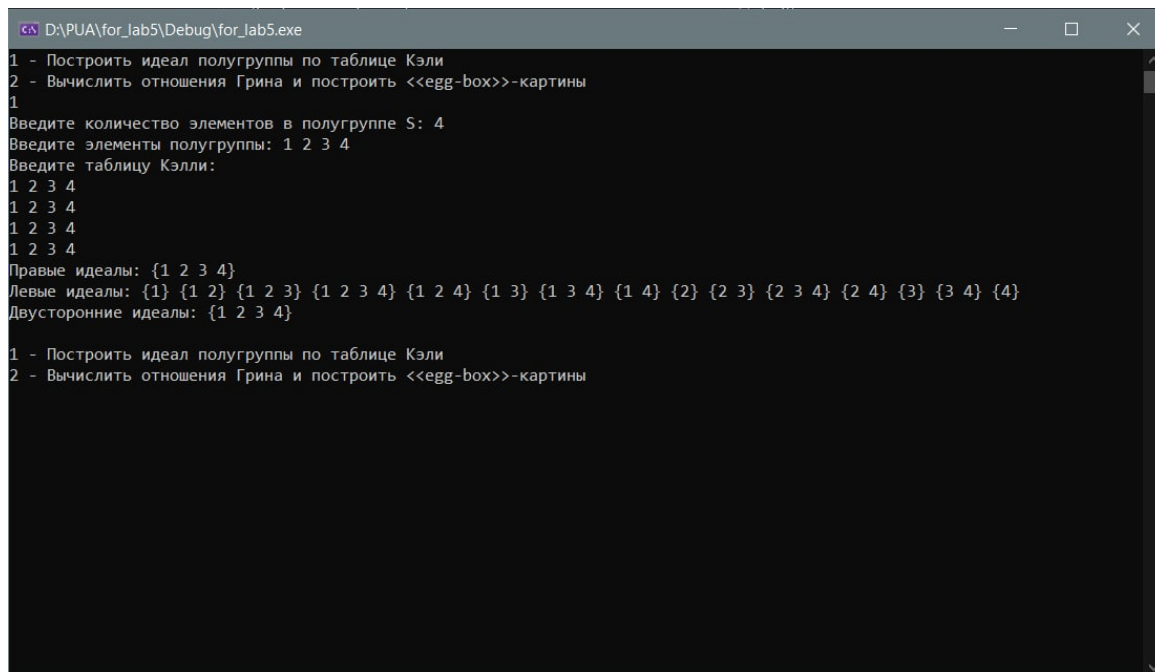
Шаг 3.3. Цикл по i от 0 до $res1.size$, по j от 0 до $res2.size$, «egg-box»-картина строится пересечением $res1_i$ и $res2_j$.

Шаг 4. Выводятся отношения Грина \mathcal{R} , \mathcal{L} , \mathcal{J} , \mathcal{H} , \mathcal{D} и «egg-box»-картины.

Трудоёмкость алгоритма $O(N^3)$.

2.3 Код программы, на основе рассмотренных алгоритмов, на языке C++

На рисунках 2-3 можно увидеть работу, реализуемой программы, по рассмотренным алгоритмам.



```
D:\PUA\for_lab5\Debug\for_lab5.exe
1 - Построить идеал подгруппы по таблице Кэли
2 - Вычислить отношения Грина и построить <<egg-box>>-картины
1
Введите количество элементов в подгруппе S: 4
Введите элементы подгруппы: 1 2 3 4
Введите таблицу Кэли:
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
Правые идеалы: {1 2 3 4}
Левые идеалы: {1} {1 2} {1 2 3} {1 2 3 4} {1 2 4} {1 3} {1 3 4} {1 4} {2} {2 3} {2 3 4} {2 4} {3} {3 4} {4}
Двусторонние идеалы: {1 2 3 4}
1 - Построить идеал подгруппы по таблице Кэли
2 - Вычислить отношения Грина и построить <<egg-box>>-картины
```

Рисунок 2

```

D:\PUA\for_lab5\Debug\for_lab5.exe
Левые идеалы: {1} {1 2} {1 2 3} {1 2 3 4} {1 2 4} {1 3} {1 3 4} {1 4} {2} {2 3} {2 3 4} {2 4} {3} {3 4} {4}
Двусторонние идеалы: {1 2 3 4}

1 - Построить идеал полугруппы по таблице Кэли
2 - Вычислить отношения Грина и построить <<egg-box>>-картины
2
Введите число элементов в полугруппе S: 4
Введите элементы полугруппы: 1 2 3 4
Введите таблицу Кэлли
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4

Отношения Грина:
R = {(1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) }
L = {(1, 1) (2, 2) (3, 3) (4, 4) }
J = {(1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) }
H = {(1, 1) (2, 2) (3, 3) (4, 4) }
D = {(1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) }

Классы эквивалентности в R:
{{1, 2, 3, 4}}

Классы эквивалентности в L:
{{1}, {2}, {3}, {4}}

Классы эквивалентности в D:
{{1, 2, 3, 4}}
<<egg-box>> - картина:
      {1}{2}{3}{4}
{1, 2, 3, 4}      {1}{2}{3}{4}

```

Рисунок 3

Листинг программы

```

#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <algorithm>
#include <iomanip>

using namespace std;

set <vector <char>> &allel;

int Find1(char k, vector<char> bunch, int n) {
    for (int i = 0; i < bunch.size(); i++)
        if (k == bunch[i])

```

```

        return i;
    }

```

```

bool Chek_ass(vector <vector <char>> res,
vector<char> bunch, int n) {
    for (int a = 0; a < n; a++)
        for (int b = a; b < n; b++)
            for (int c = 0; c < n; c++)
                if (res[Find1(res[a][b], bunch, n)][c]
!= res[a][Find1(res[b][c], bunch, n)])
                    return false;
    return true;
}

```

```

bool Find(char k, vector<char> bunch, int n) {
    for (int i = 0; i < bunch.size(); i++)
        if (k == bunch[i])
            return true;
}

```

```

bool Ideals(vector <char> a, vector <char> bunch,
vector <vector <char>> Kel, int s) {
    for (int i = 0; i < a.size(); i++)
        for (int j = 0; j < bunch.size(); j++) {
            if (s == 1) {
                if (!Find(Kel[Find1(a[i], bunch,
bunch.size())][j], a, a.size()))
                    return false;
            }
            else {
                if (!Find(Kel[j][Find1(a[i], bunch,
bunch.size())], a, a.size()))
                    return false;
            }
        }
}

```

```

    }
    return true;
}

```

```

set <vector<char>> Make(set <vector<char>> v1,
set <vector<char>> v2) {
    set <vector<char>> res;
    for (vector<char> a : v1)
    for (vector<char> b : v2)
    if (a == b)
    res.insert(a);
    return res;
}

```

```

void exit(set <vector <char>> res) {
    for (vector <char> a : res) {
        cout << "{";
        for (int j = 0; j < a.size(); j++)
            if (j == a.size() - 1)
                cout << a[j];
            else
                cout << a[j] << " ";
        cout << "} ";
    }
}

```

```

void Ex1() {
    int n, m;
    char t1;
    cout << "Enter the number of
elements in the semigroup S: ";
    cin >> n;
    vector <char> bunch;
}

```

```

cout << "Enter semigroup elements: ";
for (int i = 0; i < n; i++) {
    cin >> t1;
    bunch.push_back(t1);
}

vector <vector <char>> Kel;
Kel.resize(n);
cout << "Enter Callie table:\n";
for (int i = 0; i < n; i++) {
    Kel[i].resize(n);
    for (int j = 0; j < n; j++)
        cin >> Kel[i][j];
}

if (!Chek_ass(Kel, bunch, n)) {
    cout << "\nSemigroup associativity is
    NOT satisfied!\n";
    return;
}

for (int i = (1 << n) - 1; i >= 0; i--) {
    vector <char> res2;
    for (int j = 0; j < n; j++) {
        if (i & (1 << j)) {
            res2.push_back(bunch[j]);
        }
    }
    if (res2.size() > 0)
        allel.insert(res2);
}

set <vector <char>> res;
for (vector <char> a : allel)

```

```

    if (Ideals(a, bunch, Kel, 1))
    res.insert(a);
    cout << "Right ideals: ";
    exit(res);
    cout << endl;

    set <vector <char>> res3;
    for (vector <char> a : allel)
    if (Ideals(a, bunch, Kel, 0))
    res3.insert(a);
    cout << "Left ideals: ";
    exit(res3);
    cout << endl;

    cout << "Bilateral ideals: ";
    set <vector<char>> res4 = Make(res, res3);
    exit(res4);
    cout << endl;
}

set < pair <char, char>> Comp(set<pair <char, char>> R,
set<pair <char, char>> L) {
    set < pair <char, char>> res;
    for (pair <char, char> a : R)
    for (pair <char, char> b : L)
    if (a.second == b.first)
    res.insert(make_pair(a.first, b.second));
    return res;
}

set <pair <char, char>> Peres(set <pair <char, char>> v1,
set <pair <char, char>> v2) {
    set <pair <char, char>> res;
    for (pair <char, char> a : v1)

```

```

        for (pair <char, char> b : v2)
            if (a == b)
                res.insert(a);
        return res;
    }

vector<int> Per(vector<int> v1, vector<int> v2) {
    vector<int> res;
    for (int el1 : v1)
        for (int el2 : v2)
            if (el1 == el2)
                res.push_back(el1);
    return res;
}

set<int> Per2(vector<int> v1, vector<int> v2) {
    set<int> res;
    for (int el1 : v1)
        for (int el2 : v2)
            if (el1 == el2)
                res.insert(el1);
    return res;
}

vector<vector<int>> Equi(int** v, int n, char* els) {

    vector<vector<int>> res;
    int* count = new int[n];
    for (int i = 0; i < n; i++) {
        count[i] = 1;
    }

    for (int i = 0; i < n; i++) {
        if (count[i]) {

```

```

        vector<int> srez;
        for (int j = i; j < n; j++) {
            if (v[i][j] && count[j]) {
                count[j] = 0;
                srez.push_back(j);
            }
        }
        count[i] = 0;
        res.push_back(srez);
    }
}

cout << endl << "{";
for (int i = 0; i < res.size(); i++) {
    cout << "{";
    for (int j = 0; j < res[i].size(); j++) {
        cout << els[res[i][j]];
        if (j != res[i].size() - 1)
            cout << ", ";
    }
    cout << "}";
    if (i != res.size() - 1)
        cout << ", ";
}
cout << "}" << endl;

return res;
}

void Ex2() {
    int n, m;
    char t1;
    cout << "Enter the number of
elements in the semigroup S: ";

```



```

cin >> n;
vector <char> bunch;
cout << "Enter semigroup elements: ";
for (int i = 0; i < n; i++) {
    cin >> t1;
    bunch.push_back(t1);
}

char* a = new char[n];
for (int i = 0; i < bunch.size(); i++) {
    a[i] = bunch[i];
}

vector <vector <char>> Kel;
Kel.resize(n);
cout << "Enter Callie table\n";
for (int i = 0; i < n; i++) {
    Kel[i].resize(n);
    for (int j = 0; j < n; j++)
        cin >> Kel[i][j];
}
cout << "\nGreen's relationship:\n";
//R
set <pair <char, char>> res1;
for (int i = 0; i < bunch.size(); i++)
for (int j = 0; j < bunch.size(); j++) {
    set <char> set11;
    set11.insert(bunch[i]);
    for (int k = 0; k < Kel[i].size(); k++)
        set11.insert(Kel[i][k]);
    set <char> set12;
    set12.insert(bunch[j]);
    for (int k = 0; k < Kel[j].size(); k++)
        set12.insert(Kel[j][k]);
}

```

```

        if (set11 == set12)
            res1.insert(make_pair(bunch[i], bunch[j]));
    }
    //L
    set <pair <char, char>> res2;
    for (int i = 0; i < bunch.size(); i++)
    for (int j = 0; j < bunch.size(); j++) {
        set <char> set11;
        set11.insert(bunch[i]);
        for (int k = 0; k < n; k++)
            set11.insert(Kel[k][i]);
        set <char> set12;
        set12.insert(bunch[j]);
        for (int k = 0; k < n; k++)
            set12.insert(Kel[k][j]);
        if (set11 == set12)
            res2.insert(make_pair(bunch[i], bunch[j]));
    }
    //J
    set <pair <char, char>> res3;
    for (int i = 0; i < bunch.size(); i++)
    for (int j = 0; j < bunch.size(); j++) {
        set <char> set11;
        set11.insert(bunch[i]);
        set <char> set111;
        for (int k = 0; k < n; k++)
            set111.insert(Kel[k][i]);
        for (char r : set111) {
            for (int k = 0; k < n; k++)
                set11.insert(Kel[Find1(r, bunch,
                    bunch.size())][k]);
        }

        set <char> set12;

```

```

        set12.insert(bunch[j]);
        set <char> set112;
        for (int k = 0; k < n; k++)
            set112.insert(Kel[k][j]);
        for (char r : set112) {
            for (int k = 0; k < n; k++)
                set12.insert(Kel[Find1(r, bunch,
                    bunch.size())][k]);
        }
        if (set11 == set12)
            res3.insert(make_pair(bunch[i], bunch[j]));
    }

```

```

cout << "R = {";
    for (pair <char, char> a : res1)
        cout << "(" << a.first << ", "
        << a.second << ") ";
    cout << "}\n";

```

```

cout << "L = {";
    for (pair <char, char> a : res2)
        cout << "(" << a.first << ", "
        << a.second << ") ";
    cout << "}\n";

```

```

cout << "J = {";
    for (pair <char, char> a : res3)
        cout << "(" << a.first << ", "
        << a.second << ") ";
    cout << "}\n";

```

```

set <pair <char, char>> res4 = Peres(res1, res2);
cout << "H = {";
    for (pair <char, char> a : res4)

```

```

        cout << "(" << a.first << ", "
        << a.second << ") ";
        cout << "}\n";

set <pair <char, char>> res5 = Comp(res1, res2);
cout << "D = {";
        for (pair <char, char> a : res5)
            cout << "(" << a.first << ", "
            << a.second << ") ";
        cout << "}\n";

int** matr1;
matr1 = new int* [n];
for (int i = 0; i < n; i++) {
    matr1[i] = new int[n];
    for (int j = 0; j < n; j++) {
        matr1[i][j] = 0;
    }
}
for (pair <char, char> a : res1) {
    matr1[Find1(a.first, bunch, bunch.size())]
    [Find1(a.second, bunch, bunch.size())] = 1;
}
cout << "\nEquivalence classes in R:";
vector<vector<int>> r = Equi(matr1, n, a);

int** matr2;
matr2 = new int* [n];
for (int i = 0; i < n; i++) {
    matr2[i] = new int[n];
    for (int j = 0; j < n; j++) {
        matr2[i][j] = 0;
    }
}

```

```

for (pair <char, char> a : res2) {
    matr2[Find1(a.first, bunch, bunch.size())]
    [Find1(a.second, bunch, bunch.size())] = 1;
}
cout << "\nEquivalence classes in L:";
vector<vector<int>> l = Equi(matr2, n, a);

int** matr5;
matr5 = new int* [n];
for (int i = 0; i < n; i++) {
    matr5[i] = new int[n];
    for (int j = 0; j < n; j++) {
        matr5[i][j] = 0;
    }
}
for (pair <char, char> a : res5) {
    matr5[Find1(a.first, bunch, bunch.size())]
    [Find1(a.second, bunch, bunch.size())] = 1;
}
cout << "\nEquivalence classes in D:";
vector<vector<int>> d = Equi(matr5, n, a);

cout << "<<egg-box>> - picture:\n";
for (vector<int> cl : d) {
    vector<vector<int>> res_l;
    vector<vector<int>> res_r;
    for (vector<int> el : l) {
        if (Per(cl, el) == el)
            res_l.push_back(el);
    }
    for (vector<int> el : r)
        if (Per(cl, el) == el)
            res_r.push_back(el);
    cout << setw(15);
}

```

```

for (vector<int> ans : res_l) {
    cout << "{";
    string res = "";
    for (int x : ans) {
        res = res + a[x] + ", ";
    }
    res.pop_back();
    res.pop_back();
    res += "}";
    cout << res;
}
cout << endl;
for (vector<int> ans : res_r) {
    cout << "{";
    string res = "";
    for (int x : ans) {
        res = res + a[x] + ", ";
    }
    res.pop_back();
    res.pop_back();
    res += "}";
    cout << res << setw(14);
    for (vector<int> ans2 : res_l) {
        string res2 = "";
        res2 += "{";
        set<int> k = Per2(ans, ans2);
        for (int x : k) {
            res2 = res2 + a[x] + ", ";
        }
        res2.pop_back();
        res2.pop_back();
        res2 += "}";
        cout << res2;
    }
}

```

```

        cout << endl;
    }
    cout << endl;
    cout << endl;
}

}

int main() {
    setlocale(LC_ALL, "Russian");
    for (;;) {
        cout << "1 – Construct an ideal of
a semigroup from the Cayley tableau
\n2 – Calculate Green's ratios and
build <<egg-box>>-pictures\n";
        int x;
        cin >> x;
        switch (x) {
            case 1:
                Ex1();
                cout << endl;
                break;
            case 2:
                Ex2();
                cout << endl;
                break;
            case 0:
                break;
        }
    }
}

```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были рассмотрены понятия идеалов полугруппы, понятия и свойства отношений Грина на полугруппах, разобран алгоритм построения «egg-box»-картины конечной полугруппы. А также были реализованы алгоритм построения идеалов полугруппы по таблице Кэли, алгоритм вычисления отношений Грина и алгоритм построения «egg-box»-картины конечной полугруппы.